

**REMARKS**

Claims 1-10, 13-23, and 26-28 are pending and remain. Claims 1, 14, and 28 have been amended.

**Rejections under 35 U.S.C. § 103(a) over Schmidt, Brewer, and Chase**

- 5        Claims 1-10, 13-23, and 26-28 stand rejected under 35 U.S.C. § 103(a) as obvious over U.S. Patent No. 6,546,554, to Schmidt et al. ("Schmidt"), in view of U.S. Patent No. 6,219,787 to Brewer, and U.S. Patent No. 7,240,107 to Chase-Salerno et al. ("Chase"). Applicant traverses.

- 10        The examiner bears the initial burden of factually supporting any *prima facie* conclusion of obviousness, which includes a clear articulation of the reasons or rationale why the claimed invention would have been obvious. MPEP 2142. Exemplary rationales to support a conclusion of obviousness are listed in MPEP 2143, although the list is not all-inclusive.

- 15        The rationale based on combining prior art elements according to known methods appears to have been applied. Therefore, to establish a *prima facie* case of obviousness under this rationale, the examiner has the burden of providing the following: (1) a finding that the prior art included each element claimed, although not necessarily in a single prior art reference, with the only difference between the claimed invention and the prior art being the lack of actual combination of the
- 20        elements in a single prior art reference; (2) a finding that one of ordinary skill in the art could have combined the elements as claimed by known methods, and that in combination, each element merely performs the same function as it does separately; (3) a finding that one of ordinary skill in the art would have recognized that the results of the combination were predictable; and (4) whatever
- 25        additional findings based on the *Graham* factual inquiries may be necessary, in view of the facts of the case under consideration. MPEP 2143(A). If any of the findings cannot be made, this rationale cannot be used to support a conclusion that the claim would have been obvious. *Id.*

- 30        Claim 1 recites a checking mechanism to receive an installation predicate object comprising code from the service host system to determine availability of

the network service software on the service host system and to verify prerequisites against a runtime environment through the service host system by testing hardware and software components of the requesting system. Claim 14 recites receiving on the requesting system an installation predicate object comprising code from the service host system to determine availability of the network service software and to verify prerequisites against a runtime environment through the service host system by testing hardware and software components of the requesting system. Claim 28 recites means for receiving on the requesting system, an installation predicate object comprising code from the service host system to determine availability of the network service software on the service host system and to verify prerequisites against a runtime environment through the service host system by testing hardware and software components of the requesting system. Support for the claim amendments can be located on the specification on page 8, lines 20-31. Thus, no new matter has been entered.

The Schmidt-Brewer-Chase combination fails to teach such limitations. Schmidt and Chase were discussed in detail in the Response to Office Action of December 24, 2009. Brewer discloses securely downloading native code (Brewer, Abstract). The native code is stored with attributes and source code (Brewer, Col. 19, lines 42-48). A JAVA Bean is resident in a network server and acts as a wrapper for the native code (Brewer, Col. 19, lines 42-43 and 61-64). A host loads an applet containing the Bean, which can be queried for a size of the native code, code type, and millions of instructions per second required (Brewer, Col. 20, lines 34-43). If the intended processor has sufficient resources to run the code, the code can be installed to execute on an intended processor, including the host processor or a DSP processor (Brewer, Col. 20, lines 43-47). Therefore, Brewer teaches merely querying the Bean to determine a size of native code, code type, and MIPS required, rather than testing, by an installation predicate object, the hardware and software components of a requesting system upon which the installation predicate object is executed.

Schmidt and Chase fail to remedy the shortcoming of Brewer. Schmidt

teaches transmitting a file from a remote server to a client system (Schmidt, Col. 4, lines 27-34). A link to a JNL metafile located on the remote server is encountered and a copy of the JNL metafile is downloaded to a local temporary file on the client system (Schmidt, Col. 7, lines 58-65). A JNET helper  
5 application installed on the client system parses the JNL metafile to identify components necessary to install and launch an application described in the JNL metafile (Schmidt, Col. 7, line 58-Col. 8, line 1). Subsequently, the application is installed and launched (Schmidt, Col. 8, lines 13-21). Thus, Schmidt teaches parsing a metafile to obtain necessary components for installation of an  
10 application, rather than verifying prerequisites against a runtime environment through the service host system by testing hardware and software components of the requesting system.

Additionally, Chase teaches a method for copying an operating system in a cluster computing environment (Abstract). A server responds to a client request  
15 for installation by shutting down DHCP protocol and installing the operating system with any services or installable images required for replication on the requesting client (Col. 3, lines 49-51). Thus, Chase teaches installing services necessary for the client to replicate itself, but fails to detail how or what the services are to be installed. Therefore, Chase fails to teach an installation  
20 predicate object to verify prerequisites against a runtime environment through the service host system by testing hardware and software components of the requesting system.

Further, the Schmidt-Chase-Brewer combination is improper as each element of the respective references fail to merely perform the same function in  
25 combination as when separate. Schmidt teaches parsing metafiles to identify necessary installation components, whereas Brewer teaches loading a Bean with native code onto a host. To load the Bean, an object, such as the DSPLoader, is created (Brewer, Col. 20, lines 58-60). The DSPLoader can be local on the host or remote (Brewer, Col. 20, line 60-Col. 21, line 4). Thus, in Schmidt, the client  
30 system would need to be modified to include the DSPLoader and execute the

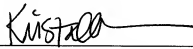
Bean, as described in Brewer, for obtaining code. Therefore, the Schmidt, Chase, and Brewer references are improperly combined.

Accordingly, a *prima facie* case of obviousness has not been shown with respect to independent Claims 1, 14, and 28. A similar conclusion would adhere under the other exemplary rationales in the KSR Guidelines to fail to demonstrate obviousness. MPEP 2143. Claims 2-10 and 13 are dependent on Claim 1 and are patentable for the above-stated reasons, and as further distinguished by the limitations therein. Claims 15-23, 26, and 27 are dependent on Claim 14 and are patentable for the above-stated reasons, and as further distinguished by the limitations therein. Withdrawal of the rejection is requested.

Claims 1-10, 13-23, and 26-28 are believed to be in condition for allowance. Entry of the foregoing amendments is requested and a Notice of Allowance is earnestly solicited. Please contact the undersigned at (206) 381-3900 regarding any questions or concerns associated with the present matter.

Respectfully submitted,

Dated: May 24, 2010

By:   
Krista A. Wittman, Esq.  
Reg. No. 59594

Cascadia Intellectual Property  
500 Union St, Suite 1005  
Seattle, WA 98101

Telephone: (206) 381-3900  
Facsimile: (206) 381-3999

OA Resp (RCE 2)